Human Activity Recognition Prediction

Sean Dunagan

June 1, 2016

Executive Summary

In this project the goal was to build a prediction model that would generate predictions for data regarding the Human Activity Recognition (HAR) dataset. The HAR dataset contains numerous data-points regarding a subject's exercise, including a data point denoting "how well" they were doing the exercise (i.e. if they were using good form/technique). Our goal was to build a prediction model that would determine how well the exercise was being done (represented by the "classe" variable in the data set) based on the other data points in the data set.

I used Principal Component Analysis to increase the amount of information that could be gathered from the relationship between the covariates in the dataset. I used a Random Forest model (rf) to generate the predictions. I chose this method as it is applicable to multi-class classification and random forests are known to have great accuracy. I tried both a random forest and a boosted model, and the random forest model had a higher accuracy rate. Cross validation was used to train and test the model and generate bounds for an out-of-set error rate.

Data Pre-Processing

The following pre-processing steps were taken to prepare the data for the creation of the model: - Declare NA vales - Remove columns with NA values - Filter Out Non-variant columns - Filter Out Irrelevant columns

The data file contained the following values which we deemed to be NA: 'NA', '#DIV/0!', and the empty string ''

A number of the columns in the data-set had NA values. I decided to remove any columns which were at least .1% NA values. This was done to remove noise in the model, and ensure that all predictors in the model would be defined for any rows in the eventual test set. As there were plenty of columns which had less than .1% of their values being NA, I decided not to impute values for any of the other columns. I also decided not to impute values for the columns which had less than .1% NA values as it would increase the complexity of the model and would likely not yield much improvement in the model.

The R method nearZeroVar() was used to remove any covariates which had a variance near zero. These covariates would not provide much information gain, and therefore do not have much to offer our prediction model. They would be more likely to create noise in the model than to increase the model's accuracy.

A number of columns did not seem like they would be relevant in terms of determining the classe of a given row. These columns were removed as they would not relate to any physical proprties of the motion of exercise. For example I removed the timestamp columns from the data-set. While it is true that consecutive seconds will likely belong to the same classe of exercise, a real-world implementation of this predictor would not want to use the timestamp as a predictor; the exact time that an exercise is done has no bearing on whether the exercise is being done with good or bad form. By this same logic I also removed the user_name column; if the model was attempting to predict the classe of exercise for a new person who had not prevously been in the dataset, the user_name data point would not be of any help and may simply introduce noise into the model.

Principal Component Analysis

Principal Component Analysis was used on the covariates in the dataset which were chosen to build the model. This was done to allow for ample information gain regarding the covariates. The tradeoff here is training-set accuracy vs overfitting and clarity. The more components which are used, the more accurate the model will be on the training set. However, this may lead to overfitting on the training set, resulting in less accuracy on the out-of-sample set. A model with many PCA components would also be more difficult to interperet or communicate. More components will also result in the model taking longer to construct.I decided to simply use the “pca” preProcess method in the R train() function.

Cross Validation

The concept behind cross validation is that we want to be able to test our prediction model on sample data before using it on the actual test set. As such, I took a portion of the training data set and set it aside to use it to test the model. This was done with

```
createDataPartition(y=relevantArmBandData$classe, p=0.8, list=FALSE);
```

It randomly assigned 80% of the data points to be used as training set points and 20% to be used as a test set.

Model Evaluation

I used the confusionMatrix() method to evaluate the prediction model’s results. The accuracy and confusion matrix are shown below:

```
## [1] "Accuracy Rate: 1"
```

```
## [1] "In-Set Error Rate: 0"
```

```
## [1] "Confusion Matrix"
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 4464    0    0    0    0
##          B    0 3038    0    0    0
##          C    0    0 2738    0    0
##          D    0    0    0 2573    0
##          E    0    0    0    0 2886
```

The accuracy shown here is considered the in-set error rate; it is the error rate we experienced regarding the data in our training set. The testing set will contain different data points than those in our training set. Since our model was built against the data in the training set, we would expect it to perform worse on a different set of data. As such, we would expect the out-of-set error rate to be greater than our in-set error rate.

There are 5 possible values for the “classe” variable. As such, random guessing would be expected to result in 20% accuracy. As such, our model is much more successful than random guessing would be.